Designing Attack Resistant Stream Ciphers using Cellular Automata

Swapan Maiti[1] and Dipanwita Roy Chowdhury[2]

[1] S. Maiti Indian Institute of Technology Kharagpur, India
swapankumar_maiti@yahoo.co.in
[2] D. Roy Chowdhury Indian Institute of Technology Kharagpur, India
drc@cse.iitkgp.ac.in

Abstract. In designing a stream cipher, Cellular Automata (CA) in particular the nonlinear CA play an important role. Wolfram identified Rule 30 as a powerful nonlinear function for cryptographic applications. However, Meier and Staffelbach mounted an attack (MS attack) on Rule 30 CA. Some of the CA based stream ciphers have shown to be resistant against popular known attacks, but none of these ciphers consider MS attack as a security threat. This paper analyzes maximum period nonlinear hybrid CA (M-NHCA) with nonlinearity injection into single and multiple inject point(s) and shows that the M-NHCA with multiple inject points is secure against MS attack. We present a design construction of a stream cipher employing both a maximum period linear hybrid CA (M-LHCA) and an M-NHCA in conjunction with a rotational symmetric bent function. The proposed cipher has also been analyzed in aspect of known popular attacks in particular, the fault attack against which most of the eStream candidates like Grain-128 are vulnerable. The use of CA gives an additional benefit of a scalable architecture. The cipher is hardware efficient which is evident from FPGA implementation.

Keywords: Stream cipher · Cryptanalysis · Cryptographic attacks · Cellular Automata · FPGA implementation

1 Introduction

A stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom keystream. In a stream cipher, each plaintext digit is encrypted one at a time with the corresponding digit of the keystream, to produce a digit of the ciphertext. Stream ciphers have gained popularity in recent years in resource constrained environments. The eSTREAM project started in 2004, introduced a number of stream ciphers in hardware and software efficient environments. The winners of the eSTREAM portfolio ciphers are 7 candidates, 4 in software category and 3 in hardware category. A number of cryptanalysis of ciphers are studied by the research community. Side Channel Attack (SCA) of stream ciphers is one class of analysis of strength of the ciphers, which includes power analysis, fault analysis and timing analysis. Fault attacks constitute one of the most interesting Side Channel Attacks. Most of the ciphers in the eStream portfolio are susceptible to fault attacks [HS04], [BMS12], [HR08]. To overcome these problems, Cellular Automata (CA) were proposed as one possible candidate to prevent attacks [KR11b].

CA are very powerful computational model. The self-evolving nature of CA has numerous applications including the generation of good pseudorandom sequences. Its high diffusion property makes CA a very attractive candidate for crypto-primitives. Rule 30 CA has long been considered a good pseudo-random generator and studied for cryptography [Wol85], [Wol86]. It passed various statistical tests for pseudo-randomness with good results, until Meier and Staffelbach proposed an attack, called MS Attack [MS91]. In literature, NOCAS [KR11b], CASca [GR15], CASTREAM [DR14], CAR30 [DR13] are all Grain-like ciphers where LFSR is replaced by Linear CA and NFSR is replaced by nonlinear CA. In all these CA based ciphers and in Grain-128 [HJMM06], a feedback path is incorporated from linear block to nonlinear block and faults introduced in the Linear block (LFSR in case of Grain-128) propagate to the nonlinear block (NFSR in case of Grain-128) and hence, it is a weakness of all these works in aspect of the fault attack. Moreover, these works did not consider the MS attack though it is a real threat against a CA based cipher.

In this work, we study nonlinear rules of M-NHCA and show that M-NHCA with nonlinear function injections into multiple inject points provide a maximum length cycle as well as better cryptographic primitives and they are also secure against MS attack. We propose a design of a stream cipher using an M-NHCA and an M-LHCA. In the proposed cipher, there is no need of any feedback path from the linear block to nonlinear block because of using both maximum period nonlinear and linear CA, and a rotational symmetric bent function which makes the mixing of contents of the linear and nonlinear CA. Hence, this cipher overcomes the weakness of fault propagation as mentioned earlier. The design of the cipher prevents known popular attacks. The security of the proposed cipher is analyzed in the light of related attacks. The main contribution of this work can be summarized as follows:

- To design an attack resistant stream cipher using cellular automata
- Introducing multiple nonlinearity injection points to the maximum period nonlinear hybrid CA (M-NHCA) to increase nonlinearity, and establishing a proof of maximum periodicity of the M-NHCA
- Security analysis of the M-NHCA against MS attack
- Detailed security analysis of the proposed stream cipher with a special emphasis to fault attacks
- Hardware implementation of the cipher on FPGA platform

The rest of this paper is organized as follows. The design architecture and working principle of the proposed cipher are shown in Section 2. Section 3 describes the design rationale of each component of the cipher. The detailed security analysis is furnished in Section 4. The robustness of the cipher against the existing cryptanalysis techniques with a special focus on MS attack is also studied in detail in this section. The hardware implementation results on Xilinx Spartan 3 XC3S200-4FT256 FPGA platform are depicted in Section 5. Finally, the paper is concluded in Section 6.

2 Design Architecture

A new stream cipher using maximum period linear and nonlinear CA has been briefly introduced in [MGR17]. In this work, we present the details of the security analysis and show how CA can be used as a better crypto-primitive in designing an attack resistant stream cipher. The cipher is also shown to be hardware efficient when implemented on FPGA. The cipher consists of three building blocks, a Linear Hybrid Cellular Automata (LHCA), a Nonlinear Hybrid Cellular Automata (NHCA) and a final combiner function $h(\cdot)$. The overview of the design can be found in Fig. 1. In the following subsections, the workings of all the building blocks are illustrated in detail along with a discussion about the scalable architecture of the design. Finally, we describe how the cipher must be initialized with the key and the initialization vector (IV).

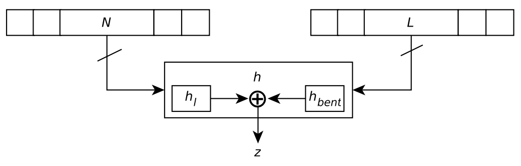


Figure 1: Overview of the design of the cipher

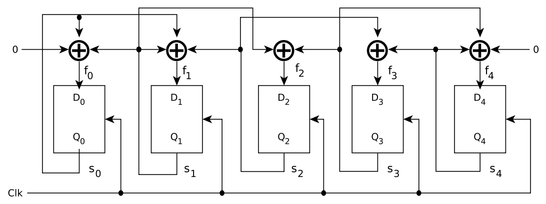


Figure 2: M-LHCA with rule vector [1, 1, 0, 0, 1]

2.1 Linear Block

This block uses a 128-bit maximum period LHCA (M-LHCA) $\mathcal{L}$ denoted by $\{s_0, s_1, \cdots, s_{127}\}$ where s_i denotes the state of the i^{th} cell of $\mathcal{L}$. A primary knowledge of cellular automata is required for the remainder of the paper. So before delving into further details, we start by briefly discussing some basic notions of CA.

A cellular automata is a linear finite state machine that consists of an array of n-cells represented by $\{s_0, s_1, \cdots, s_{n-1}\}$, each cell capable of storing a single bit. If the state of a cell at a given instant is dependent upon the neighboring cells including itself, then it is called a 3-neighborhood CA. However, out of all possible Boolean functions, called rules, only two are of prime interest i.e. Rules 90 and 150 (ascertained from the decimal value of their position in the truth table). The state transition function of the i^{th} cell can be expressed as:

$$f_i = s_{i-1} \oplus d_i.s_i \oplus s_{i+1},\ d_i = \begin{cases} 0, & \text{if } s_i \text{ follows Rule 90} \\ 1, & \text{if } s_i \text{ follows Rule 150.} \end{cases}$$

Thus, an LHCA can be completely specified by a combination of Rules 90 and 150, denoted as an n-tuple $[d_0, d_1, \cdots, d_{n-1}]$. For example, a 5-cell M-LHCA with rule vector $[1, 1, 0, 0, 1]$ is shown in Fig. 2. Further details of CA can be found in [CRNC97].

The M-LHCA $\mathcal{L}$ used in the design is selected in a way to ensure maximum periodicity. This is accomplished by exploiting an important result [CRNC97], where a one-to-one correspondence between a maximum period LHCA and a primitive polynomial was proved. Also, in another pioneering work [CM96] on CA, an algorithm for synthesizing an LHCA from a given irreducible polynomial was presented by Cattell and Muzio. As primitive polynomials are also irreducible in nature, this effectively reduces the problem of finding a maximum length LHCA to that of selecting a readily available primitive polynomial. In our work, the characteristic polynomial of $\mathcal{L}$, denoted by $f(x)$, is a primitive polynomial defined as:

$$f(x) = x^{128} + x^{29} + x^{27} + x^2 + 1$$

The rule value of the M-LHCA $\mathcal{L}$ synthesized from $f(x)$, a primitive polynomial of degree 128, is given as $0x48882FBD67031A7A7A79C0E6BDF41112$. For the sake of simplicity, the rule value of the CA is given in hexadecimal notation i.e. a CA rule value 0xA5 denotes the rule vector $[1, 0, 1, 0, 0, 1, 0, 1]$. In the following subsection, the nonlinear component is discussed in detail.

2.2 Nonlinear Block

This block uses a 128-bit maximum period Nonlinear Hybrid Cellular Automata (M-NHCA) $\mathcal{N}$ represented by $\{b_0, b_1, \cdots, b_{127}\}$ which is synthesized from a 128-bit maximum period Linear Hybrid Cellular Automata (M-LHCA). A synthesis algorithm is explained in [GSSR14], which produces a synthesized M-NHCA from an M-LHCA with nonlinearity injection at a single injection point. In this work, we remove the restriction of single injection point, and introduce multiple injection points to increase nonlinearity of the synthesized M-NHCA. A preliminary concept of injecting nonlinearity at multiple points is briefly introduced in [MC18]. In this work, we explore this method along with the analytical behavior of the CA with multiple nonlinearity injection points. The following section presents the synthesis of the NHCA with an example in detail.

2.2.1 M-NHCA with Nonlinearity Injection into Multiple Inject Points

The following algorithm is an NHCA Synthesize Algorithm for nonlinearity injection with multiple inject positions.

Algorithm 1 NHCA Synthesize Algorithm with Multiple Inject Points

Input: An n-bit maximum period linear hybrid CA with ruleset $\mathcal{F}_L$; A set of m positions $\{i_1, i_2, \cdots, i_m\}$ to inject nonlinear functions, where $1 < i_1 < i_2 < \cdots < i_m < n-2$ and $|j-k| \geq 4$ for j, $k \in \{i_1, i_2, \cdots, i_m\}$; The set $\mathcal{S}$ of cells of the LHCA

Output: A maximum period NHCA ruleset $\mathcal{F}_N$

```
1. $\mathcal{F}_N \leftarrow \mathcal{F}_L$
2. Let $\mathcal{F}_N = \{f_{n-1}, \cdots, f_0\}$   ▷ Cell updated functions
3. For all $j \in \{i_1, i_2, \cdots, i_m\}$
4.     $\mathcal{X} \subset \mathcal{S} : \forall x \in \mathcal{X}, x \notin \mathbf{N}(j)$   ▷ Select a subset from $\mathcal{S}$, $\mathbf{N}(j)$ denotes the neighbor
   set of $j^{th}$ cell i.e., the cells in positions $j-1$, $j$, $j+1$
5.     $P \leftarrow \mathbf{f_N}(\mathcal{X})$   ▷ $\mathbf{f_N}$ is a nonlinear function
6.     $f_j \leftarrow f_j \oplus P$   ▷ Inject P into the $j^{th}$ cell
7.     $(f_j \rightarrow^P f_{j\mp1})$   ▷ Apply shifting operation from $(j-1)^{th}$ to $(j+1)^{th}$ cells
8.     $f_j \leftarrow f_j \oplus P$   ▷ Inject updated P into the $j^{th}$ cell
   End For
9. **Return** $\mathcal{F}_N$
```

Let an n-bit M-NHCA be synthesized from an n-bit M-LHCA of the rule vector $[d_0, d_1, \cdots, d_{n-1}]$, and with nonlinearity injections into m number of inject points denoted by i_1, i_2, $\cdots$, i_m, where

$$d_i = \begin{cases} 0, & \text{if } i^{th} \text{ cell follows Rule 90} \\ 1, & \text{if } i^{th} \text{ cell follows Rule 150.} \end{cases}$$

Let the synthesized M-NHCA be represented by $\{b_0, b_1, \cdots, b_{n-1}\}$, where b_i denotes the state of the i^{th} cell of M-NHCA and m number of cells for m inject points be denoted by b_{i_1}, b_{i_2}, $\cdots$, b_{i_m}, where $1 < i_1 < i_2 < \cdots < i_m < n-2$ and $|k-l| \geq 4$ for k, l $\in \{i_1, i_2, \cdots, i_m\}$.

By the synthesis algorithm, we consider that a nonlinear function $f_N(\mathcal{X})$ is injected into the j^{th} cell (Algo 1, step 6), where $\mathcal{X} = \{b_{j-2}, b_{j+2}\}$ (Algo 1, step 4) and $f_N(\mathcal{X}) = (b_{j-2}.b_{j+2})$ (Algo 1, step 5). Therefore, the state transition function of j^{th} cell is updated as

$$f_j = b_{j-1} \oplus d_j.b_j \oplus b_{j+1} \oplus f_N(b_{j-2}, b_{j+2})$$

By the shifting operation (Algo 1, step 7), the state transition functions of $(j-1)^{th}$, j^{th}, $(j+1)^{th}$ cells are updated as

$$\begin{aligned} f_{j-1} &= b_{j-2} \oplus d_{j-1}.b_{j-1} \oplus b_j \oplus f_N(b_{j-2}, b_{j+2}) \\ f_j &= b_{j-1} \oplus d_j.b_j \oplus b_{j+1} \oplus f_N(b_{j-2}, b_{j+2}) \oplus (d_j+1).f_N(b_{j-2}, b_{j+2}) \\ f_{j+1} &= b_j \oplus d_{j+1}.b_{j+1} \oplus b_{j+2} \oplus f_N(b_{j-2}, b_{j+2}) \end{aligned}$$

Finally, the state transition function of j^{th} cell is updated (Algo 1, step 8) as

$$f_j = b_{j-1} \oplus d_j.b_j \oplus b_{j+1} \oplus d_j.f_N(b_{j-2}, b_{j+2}) \oplus f_N(f_{j-2}, f_{j+2})$$

where, $f_N(f_{j-2}, f_{j+2}) = (f_{j-2}.f_{j+2})$ is the updated function-value of the injected nonlinear function. Here, f_{j-2}, f_{j+2} denote the updated values of $(j-2)^{th}$ and $(j+2)^{th}$ cells respectively (Algo 1, step 1), that is,

$$\begin{aligned} f_{j-2} &= b_{j-3} \oplus d_{j-2}.b_{j-2} \oplus b_{j-1} \\ f_{j+2} &= b_{j+1} \oplus d_{j+2}.b_{j+2} \oplus b_{j+3} \end{aligned}$$

After synthesis (Algo 1, step 9), the update functions (nonlinear) of $(j-1)^{th}$, j^{th}, $(j+1)^{th}$ cells of the synthesized M-NHCA are as follows:

$$\begin{aligned} f_{j-1} &= b_{j-2} \oplus d_{j-1}.b_{j-1} \oplus b_j \oplus f_N(b_{j-2}, b_{j+2}) \\ f_j &= b_{j-1} \oplus d_j.b_j \oplus b_{j+1} \oplus d_j.f_N(b_{j-2}, b_{j+2}) \oplus f_N(f_{j-2}, f_{j+2}) \\ f_{j+1} &= b_j \oplus d_{j+1}.b_{j+1} \oplus b_{j+2} \oplus f_N(b_{j-2}, b_{j+2}) \end{aligned}$$

for all $j \in \{i_1, i_2, \cdots, i_m\}$. The update functions (linear) of all other cells of the synthesized M-NHCA are as underlying M-LHCA (Algo 1, step 1).

For multiple injection points, we consider the following criteria:

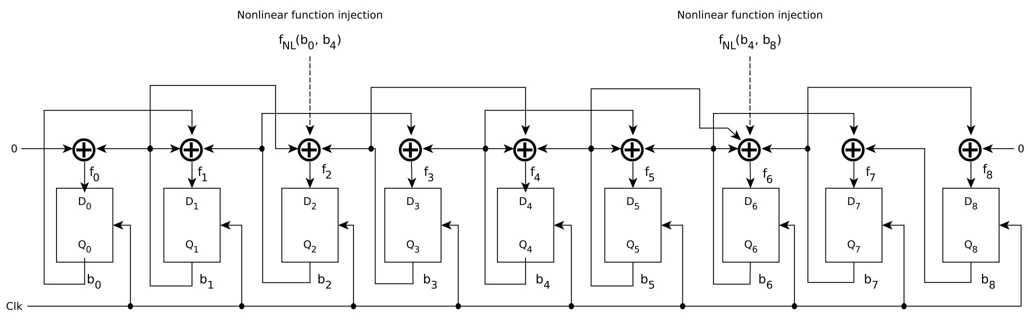
1. Nonlinear functions can be injected in cell position i, $2 \le i \le n-3$ such that the injected nonlinear function $f_N(b_{i-2}, b_{i+2}) = (b_{i-2} \cdot b_{i+2})$ can be formed properly.

2. To retain the maximum length cycle, there must be at least three cells in between any two inject positions; that is, if j and k be two inject positions then there must be $|k-l| \ge 4$; otherwise, the neighboring cells in between two inject positions will be affected simultaneously by both injections.

Algorithm 1 to synthesize an M-NHCA is explained with the following example which clearly illustrates how an M-NHCA can be synthesized by injecting nonlinear functions into two selected positions of an M-LHCA.

Example 1. Let us consider a 9-bit M-LHCA $\mathcal{L}'$ denoted by $\{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\}$ depicted in Fig. 3 with rule vector [0, 1, 0, 0, 1, 1, 1, 0, 0]. The state transition functions of the M-LHCA are as follows:

$$\begin{array}{lll} f_0 = b_1 & f_3 = b_2 \oplus b_4 & f_6 = b_5 \oplus b_6 \oplus b_7 \\ f_1 = b_0 \oplus b_1 \oplus b_2 & f_4 = b_3 \oplus b_4 \oplus b_5 & f_7 = b_6 \oplus b_8 \\ f_2 = b_1 \oplus b_3 & f_5 = b_4 \oplus b_5 \oplus b_6 & f_8 = b_7 \end{array}$$



Nonlinear function injection
$f_{NL}(b_0, b_4)$
Nonlinear function injection
$f_{NL}(b_4, b_8)$
0
0
Clk
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 Q_8
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8

Figure 3: Nonlinear function injection at two cell positions of an M-LHCA $\mathcal{L}'$

where b_i is the current state and f_i is the next state of the i^{th} cell of $\mathcal{L}'$. Let the nonlinear function $f_N(b_0, b_4) = (b_0 \cdot b_4)$ be injected at position 2, and the nonlinear function $f_N(b_4, b_8) = (b_4 \cdot b_8)$ be injected at position 6 (Algorithm 1, step 6). By Algorithm 1, the state transition functions of the synthesized M-NHCA $\mathcal{N}'$ are as follows:

$$
\begin{aligned}
f_0 &= b_1 \\
f_1 &= b_0 \oplus b_1 \oplus b_2 \oplus (b_0.b_4) \\
f_2 &= b_1 \oplus b_3 \oplus (b_1.(b_3 \oplus b_4 \oplus b_5)) \\
f_3 &= b_2 \oplus b_4 \oplus (b_0.b_4) \\
f_4 &= b_3 \oplus b_4 \oplus b_5
\end{aligned}
\qquad
\begin{aligned}
f_5 &= b_4 \oplus b_5 \oplus b_6 \oplus (b_4.b_8) \\
f_6 &= b_5 \oplus b_6 \oplus b_7 \oplus (b_4.b_8) \oplus ((b_3 \oplus b_4 \oplus b_5).b_7) \\
f_7 &= b_6 \oplus b_8 \oplus (b_4.b_8) \\
f_8 &= b_7
\end{aligned}
$$

The following theorems show that nonlinear function injections into single/multiple inject position(s) of an M-LHCA generate a maximum period NHCA (M-NHCA).

Theorem 1. *Nonlinear function injection into a single inject position of a 3-neighborhood 90/150 maximum period linear hybrid cellular automata (M-LHCA) generates a maximum period nonlinear hybrid cellular automata (M-NHCA).*

Proof. Let $\mathcal{L}$ be an n-cell 3-neighborhood null boundary maximum period LHCA denoted by $\{x_0, x_1, \cdots, x_{n-1}\}$ with rule vector $[d_0, d_1, \cdots, d_{n-1}]$, where, $d_j = 0$ (Rule 90) / 1 (Rule 150), $0 \le j \le n-1$, and the state transition function of j^{th} cell for all j, $0 \le j \le n-1$, is defined as:

$$f_j = x_{j-1} \oplus d_j.x_j \oplus x_{j+1}$$

According to the Algorithm 1, if a nonlinear function injection is made into the i^{th} cell of underlying LHCA $\mathcal{L}$, $1 < i < n-2$, with a nonlinear function

$$f_N(x_{i-2}, x_{i+2}) = (x_{i-2} \cdot x_{i+2})$$

then finally, all cells of synthesized NHCA except the neighboring cells of i, follow 90/150 rule vector of underlying LHCA (Algo. 1, step 1) and the updated state transition functions (nonlinear) of $(i-1)^{th}, i^{th}, (i+1)^{th}$ cells of synthesized NHCA are as follows (Algo. 1, step 9):

$$
\begin{aligned}
f_{i-1} &= x_{i-2} \oplus d_{i-1}.x_{i-1} \oplus x_i \oplus f_N(x_{i-2}, x_{i+2}) \\
f_i &= x_{i-1} \oplus d_i.x_i \oplus x_{i+1} \oplus d_i.f_N(x_{i-2}, x_{i+2}) \\
&\quad \oplus f_N(f_{i-2}, f_{i+2}) \\
f_{i+1} &= x_i \oplus d_{i+1}.x_{i+1} \oplus x_{i+2} \oplus f_N(x_{i-2}, x_{i+2})
\end{aligned}
$$

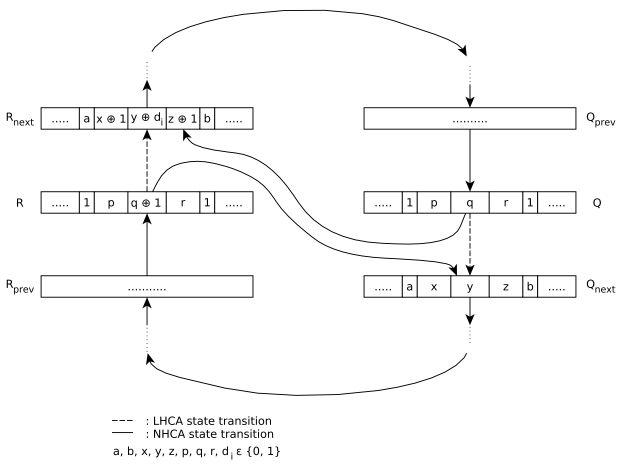

R_{next}
..... a x ⊕ 1 y ⊕ d_i z ⊕ 1 b .....
Q_{prev}
R
..... 1 p q ⊕ 1 r 1 .....
..... 1 p q r 1 .....
Q
R_{prev}
Q_{next}
..... a x y z b .....
--- : LHCA state transition
—— : NHCA state transition
a, b, x, y, z, p, q, r, d_i ε {0, 1}

Figure 4: Effect of nonlinear function injection by shifting operation

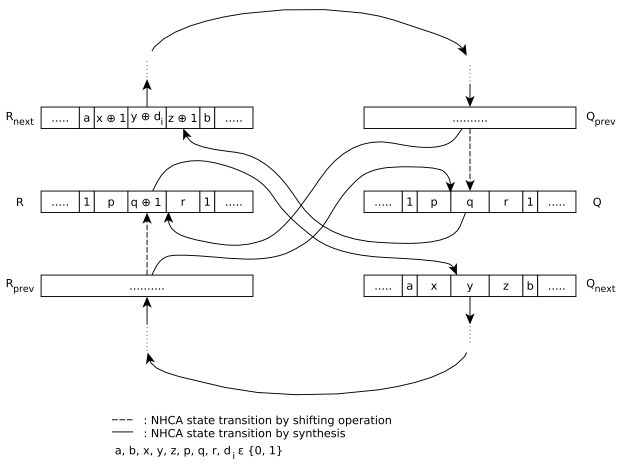

R_{next}
..... a x ⊕ 1 y ⊕ d_i z ⊕ 1 b .....
Q_{prev}
R
..... 1 p q ⊕ 1 r 1 .....
..... 1 p q r 1 .....
Q
R_{prev}
Q_{next}
..... a x y z b .....
--- : NHCA state transition by shifting operation
—— : NHCA state transition by synthesis
a, b, x, y, z, p, q, r, d_i ε {0, 1}

Figure 5: Effect of nonlinear function injection by synthesis

Let Q be a state in the maximum length cycle of underlying LHCA and Q_{next} be the next state of Q. In the state Q (ref. Fig. 4), let the bit values of the bit vector positions $\langle i-2, i-1, i, i+1, i+2\rangle$ be denoted by $\langle 1, p, q, r, 1\rangle$, where p, q, $r \in \{0, 1\}$. After shifting operation, non-zero value of the injected nonlinear function (i.e. $f_N(1, 1) = 1$) changes the truth values of $\langle x, y, z\rangle$ in the bit vector positions $\langle i-1, i, i+1\rangle$ of the state Q_{next}. Therefore, it reaches a state R_{next} in the maximum length cycle of underlying LHCA. The previous state of R_{next}, denoted by R surely contains the same bit values with that of the state Q except the content of the i^{th} cell. The contents of the i^{th} cell of Q and R are complement each other. At the end of shifting operation, non-zero value of the injected nonlinear function changes the next states of Q and R and hence, the next states Q_{next} and R_{next} are interchanged. That means the function $f_N(1, 1)$ for the pair (Q, R) changes their next states to (R_{next}, Q_{next}). Therefore, the maximum length cycle of the underlying LHCA *splits* up into two cycles (ref. Fig. 4), where Q lies in one cycle and R lies in another cycle. Hence, the shifting operation splits up the maximum length cycle of underlying LHCA into a number of cycles for concerning all nonzero states (i.e. $2^n - 1$

states) and every nonzero state lies in any one cycle.

Let Q_{prev} and R_{prev} be the previous states of Q and R, respectively in the state transition graph after shifting operations (Fig. 5), that means the pair (Q, R) is a reachable pair from (Q_{prev}, R_{prev}). The contents of the i^{th} cell of Q and R are complement each other and the contents of the $(i-2)^{th}$ and $(i+2)^{th}$ cells of Q and R are 1's. By synthesis (Algo. 1, step 8), the nonlinear function $f_N(f_{i-2}, f_{i+2}) = f_N(1, 1)$ injected on the i^{th} cell of the states Q and R evaluates to 1, and therefore, previous states of Q and R $(i.e. Q_{prev}, R_{prev})$ are interchanged, and hence two cycles are *joined* (Fig. 5).

Every (Q, R)-like pair makes interchange their next states by shifting operations (Algo. 1, step 7), and interchange their previous states in the state transition graph by synthesis (Algo. 1, step 8). Therefore, at the end of synthesis all cycles are joined into a one cycle of all nonzero states (i.e. $2^n - 1$ states), which is a maximum length cycle. □

Theorem 2. *Nonlinear function injection into multiple inject positions of a 3-neighborhood 90/150 maximum period linear hybrid cellular automata (M-LHCA) generates a maximum period nonlinear hybrid cellular automata (M-NHCA).*

Proof. We prove the theorem by mathematical induction. Let $\mathcal{L}$ be an n-cell 3-neighborhood null boundary maximum length (i.e. $2^n - 1$) cycle LHCA denoted by $\{x_0, x_1, \cdots, x_{n-1}\}$ with rule vector $[d_0, d_1, \cdots, d_{n-1}]$, where, $d_j = 0$ (Rule 90) / 1 (Rule 150), $0 \leq j \leq n-1$, and the state transition function of j^{th} cell is defined as:

$$f_j = x_{j-1} \oplus d_j.x_j \oplus x_{j+1}$$

Let nonlinear function be injected into m no. of cells, x_{i_1}, x_{i_2}, $\cdots$, x_{i_m} of underlying LHCA $\mathcal{L}$ such that $1 < i_1 < i_2 < \cdots < i_m < n-2$ and $|j-k| \geq 4$ for $j, k \in \{i_1, i_2, \cdots, i_m\}$.

Basis step: After 1st injection into cell x_{i_1}, the synthesized NHCA produces a maximum length cycle C_1 by Theorem 1. In cycle C_1, the cells of positions $i_1 - 1$ to $i_1 + 1$ are updated by 90/150 rules (i.e. linear updation by Algo 1, step 1) as well as by nonlinear function (i.e. nonlinear updation by Algo 1, steps 6, 7, 8), but the cells of positions $i_2 - 2$ to $i_2 + 2$ are only updated by 90/150 rules (Algo 1, step 1) of underlying LHCA and in C_1, the 1st injection on cell x_{i_1} does not affect (i.e. nonlinear updation) the cells of positions $i_2 - 2$ to $i_2 + 2$ because of $|i_1 - i_2| \geq 4$. Therefore, after nonlinear function injection into x_{i_2}, the generated cycle C_2 is also a maximum length cycle by Theorem 1.

Inductive hypothesis: Suppose, the synthesized NHCA produces a maximum length cycle C_k after nonlinear function injections into k no. of cells x_{i_1}, x_{i_2}, $\cdots$, x_{i_k}, where $k < m$.

Inductive step: In cycle C_k, all injections into x_{i_1}, x_{i_2}, $\cdots$, x_{i_k} do not affect (i.e. nonlinear updation) the cells of positions $i_{k+1} - 2$ to $i_{k+1} + 2$. C_k is a maximum length cycle by induction hypothesis. After nonlinear function injection into cell $x_{i_{k+1}}$ produces a cycle C_{k+1} which is also a maximum length cycle by Theorem 1. Therefore, for multiple nonlinear function injections into m cells of underlying LHCA produces a synthesized NHCA of a maximum length cycle. □

2.2.2 Nonlinearity with Iterations.

Nonlinearity is a cryptographic property of a Boolean function. The minimum of the Hamming distances between a Boolean function and all affine functions involving its input variables is known as the nonlinearity of the function [CS09], [MVV97].

Nonlinearity of M-NHCA increases more with iterations by injecting nonlinear function in multiple inject points than single inject point. Details of computing nonlinearity of some synthesized M-NHCA with single and multiple inject point(s) are presented in Table 1. The underlying M-LHCA is synthesized [CM96] from a primitive polynomial represented as a listing of non-zero coefficients. For example, the set (9, 4, 0) in the 1st column of

Table 1 represents the polynomial $x^9 + x^4 + 1$ (primitive) for a 9-bit M-LHCA denoted by $\langle b_0, \cdots, b_8 \rangle$. The set (i, j, k) in the 2nd column represents that nonlinear function is injected in i^{th}, j^{th} and k^{th} cell positions simultaneously. Table 1 clearly illustrates that the nonlinearity of M-NHCA increases more in multiple inject points than single inject point.

Table 1: Nonlinearity of different M-NHCA with iterations

M-LHCA Polynomial (primitive)	Nty Inject position(s)	CA cell for Nty	Nty with iterations						
			1	2	3	4	5	6	7
9, 4, 0	2	b_2	4	16	64	128	128	128	128
	2, 6	b_2	4	32	128	192	192	128	192
10, 3, 0	3	b_3	48	8	64	128	128	256	256
	3, 7	b_3	48	16	64	192	256	256	384
11, 9, 8, 3, 0	3	b_3	48	32	32	256	512	64	256
	3, 7	b_3	48	64	32	256	768	64	256
12, 7, 4, 3, 0	3	b_3	48	64	64	32	32	512	512
	3, 8	b_3	48	64	128	32	48	768	768
14, 12, 11, 1, 0	5	b_5	48	32	512	512	1024	1024	1024
	5, 9	b_5	48	64	1024	512	1024	1024	2048
16, 5, 3, 2, 0	5	b_5	48	32	512	512	512	1024	1024
	5, 9	b_5	48	64	1024	512	1024	1024	1024
32, 28, 27, 1, 0	11	b_{11}	16	64	512	2048	2048	3072	3072
	7, 11, 15, 19	b_{11}	16	256	2048	3072	4096	4096	4096

Nty ≡ Nonlinearity

In the proposed cipher, the nonlinear block is designed with a synthesized M-NHCA $\mathcal{N}$. For synthesis, any 128-bit M-LHCA with 90/150 rule vectors can be taken as the underlying M-LHCA. In our design, the M-NHCA $\mathcal{N}$ is synthesized from the same M-LHCA (as described in Section 2.1) with multiple injection points. The set of injection points, denoted as $\mathcal{I}$, is as follows:

$$\mathcal{I} = \{13, 17, 29, 33, 44, 48, 64, 68, 77, 81, 93, 97, 109, 113\}.$$

Finally, the next subsection describes the details of the combiner function.

2.3 Combiner Function h(·)

The function $h(\cdot)$ can be expressed as a combination of two parts, a linear function $h_l(\cdot)$ and a bent function $h_{bent}(\cdot)$, as follows:

$$h(\cdot) = h_{bent}(\cdot) \oplus h_l(\cdot)$$

Before stating the specification of $h(\cdot)$ used in the proposed cipher, we furnish some basic concepts of a rotational symmetric Boolean function [CS09].

Let $\{x_0, x_1, \cdots, x_n\}$ be the set of input bits to a Boolean function $f(\cdot)$. For $0 \le j \le n-1$, we define the rotational shifting operation as :

$$\rho^j(x_i) = x_{(i+j)mod(n)}$$

This definition can be extended for a Boolean function $f_s(\cdot)$ as follows:

$$\rho^j(f_s(x_0, x_1, \cdots, x_{n-1})) = f_s(\rho^j(x_0), \cdots, \rho^j(x_{n-1}))$$

A rotational symmetric function is a defined as the summation of all the rotationally permuted terms of a base element which is called the short ANF. The following equation shows the expression of a rotational symmetric function:

$$f(x_0, x_1, \cdots, x_{n-1}) = \sum_{j=0}^{n-1} \rho^j(f_s(x_0, x_1, \cdots, x_{n-1}))$$

where, $f_s(\cdot)$ is the short ANF of $f(\cdot)$. For example, the function $f(x_0, x_1, x_2) = x_0x_1 \oplus x_1x_2 \oplus x_2x_0$ can be denoted as

$$f(x_0, x_1, x_2) = \sum_{j=0}^{2} \rho^j(f_s(x_0, x_1, x_2))$$

where, $f_s(x_0, x_1, x_2) = x_0x_1$.

As mentioned earlier, the function $h_{bent}(\cdot)$ in the cipher is rotational symmetric. The short ANF form of $h_{bent}(\cdot)$ is denoted as $h_s(\cdot)$. The specifications of $h_l(\cdot)$ and $h_{bent}(\cdot)$ are shown in the following equations.

$$\begin{aligned} h_s(v_0, v_1, \cdots, v_7) &= v_0v_1 \oplus v_0v_2 \oplus v_0v_3 \oplus v_0v_1v_2 \oplus v_0v_1v_4 \oplus v_0v_1v_6 \\ &\oplus v_0v_2v_4 \oplus v_0v_1v_2v_3 \oplus v_0v_1v_3v_4 \oplus v_0v_1v_3v_5 \\ h_{bent}(v_0, v_1, \cdots, v_7) &= \sum_{j=0}^{7} \rho^j(h_s(v_0, v_1, \cdots, v_7)) \\ h_l(u_0, u_1, \cdots, u_4) &= \sum_{i=0}^{4} u_i \end{aligned}$$

The 256 memory elements in the two CA represent the state of the cipher. From this state, 13 variables are taken as input to the combiner function $h(\cdot)$. Six inputs are taken from $\mathcal{L}$ and seven inputs are taken from $\mathcal{N}$. The set of the input bits, also called tap bits, corresponds to the set denoted by $\mathcal{T}$ as follows:

$$\mathcal{T} = \{s_{12}, s_{35}, s_{58}, s_{78}, s_{97}, s_{119}, b_{16}, b_{32}, b_{47}, b_{67}, b_{80}, b_{96}, b_{112}\}$$

Hence, the output function (i.e. the combiner function $h(\cdot)$) is defined as

$$z = h(v_0, v_1, \cdots, v_7, u_0, u_1, \cdots, u_4) = h_{bent}(v_0, v_1, \cdots, v_7) \oplus h_l(u_0, u_1, \cdots, u_4)$$

where, $v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7$ correspond to b_{16}, s_{12},$s_{35}, s_{58}, s_{78}, s_{97}, s_{119}, b_{96}$ and u_0, u_1, u_2, u_3, u_4 correspond to $b_{32}, b_{47}, b_{67}, b_{80}, b_{112}$.

2.4 Scalability

The design of the cipher can be scaled to any security parameter. Each primitive of the cipher is scalable as follows:

1. A primitive polynomial in GF(2) of any degree is readily available in contemporary literature. Thus, a maximum period sequence generator using linear cellular automata can easily be synthesized for a given primitive polynomial.

2. For different security parameters, the M-NHCA can be resynthesized following the Algorithm 1.

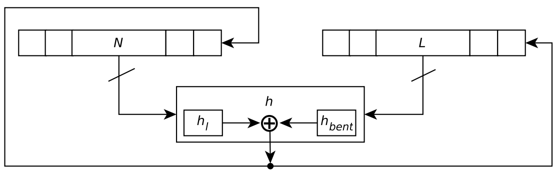


Figure 6: Initialization of the cipher

3. The final combiner function, $h(\cdot)$, should be changed for different key lengths (e.g. 128, 192, 256). Moreover, the rotational symmetric bent function $h_{bent}(\cdot)$ should also be redesigned. However, this can be done for any number of input bits with a very little computation.

Thus, the overall design is easily scalable. The only precomputation is required for synthesizing an M-NHCA and finding a suitable $h_{bent}(\cdot)$ function.

2.5 Initialization and Key Setup

Before generating any keystream, the cipher must be initialized with a key and an initialization vector (IV). Here we have used a 128-bit key k and a 128-bit IV. To initialize the cipher, the key is loaded into $\mathcal{N}$ and the IV is loaded into $\mathcal{L}$. The M-LHCA used is synthesized from a primitive polynomial and it provides maximum periodicity. Therefore, the M-LHCA state bits never contain all 0's while running the cipher and it ensures to resist the known IV attack. It overcomes the restriction of Grain of keeping 16 LSBs to be all 1's. The diffusion rate of M-NHCA evolution is much faster than that of the nonlinear block of Grain-128. In 128 clock cycles, all 256 state bits (128 bits of nonlinear block and 128 bits of linear block) will be diffused in all 256 bits, which strengthens the security against attacks like algebraic attack and fault attack etc. Therefore, the cipher is clocked for 128 cycles without producing any keystream and the output of $h(\cdot)$ is fed back and XORed with the LSBs of both $\mathcal{L}$ and $\mathcal{N}$. Thus, the initialization phase is made two times faster than that of Grain-128. The initialization phase is depicted in Fig. 6.

2.6 Cipher Design

The architectural view of the proposed cipher is shown in Fig. 7.
Linear block: it uses a M-LHCA $\mathcal{L}$ of CA polynomial $f(x) = x^{128} + x^{29} + x^{27} + x^2 + 1$.
Nonlinear block: It uses a M-NHCA $\mathcal{N}$ synthesized from $\mathcal{L}$ as described in Section 2.2.
Linear function $h_l(\cdot)$: It uses five inputs from the M-NHCA $\mathcal{N}$.
Nonlinear function $h_{bent}(\cdot)$: It uses two inputs from the M-NHCA $\mathcal{N}$ and six inputs from the M-LHCA $\mathcal{L}$ as described in Section 2.3.
Combiner function $h(\cdot)$: It combines the two functions $h_l(\cdot)$ and $h_{bent}(\cdot)$, and finally produces the cipher output.

3 Design Rationale

This section shows how the proper choices of the design parameters provide better security of the cipher.

Maximum period LHCA. In stream ciphers, linear sequence generators are added to the design to serve two purposes, firstly to provide balancedness and secondly to preclude any possibilities of power attack. Commonly this is realized using linear feedback shift

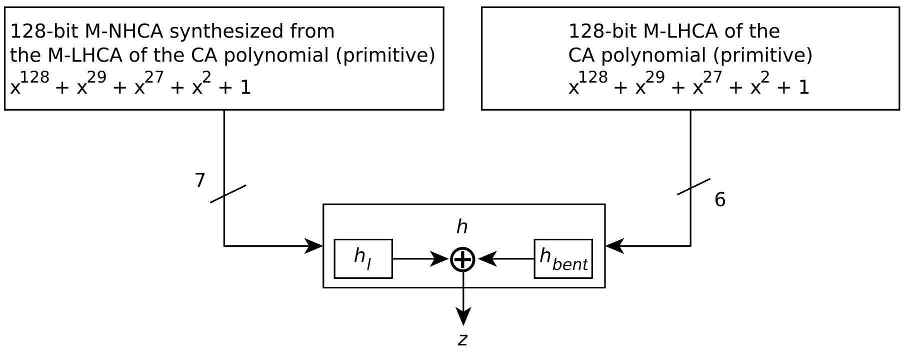

128-bit M-NHCA synthesized from the M-LHCA of the CA polynomial (primitive) $x^{128} + x^{29} + x^{27} + x^2 + 1$
128-bit M-LHCA of the CA polynomial (primitive) $x^{128} + x^{29} + x^{27} + x^2 + 1$
7
6
h
h_l
h_{bent}
z

Figure 7: Structure of the cipher

registers (LFSR). However, in our cipher, the linear sequence generator is constructed from M-LHCA instead of LFSR. This design choice is attributed to the suitability of M-LHCA as a pseudorandom number generator (PRNG) due to its better randomness property than that of the LFSR [CRNC97].

Maximum period NHCA. In case of ciphers like Grain, the faults injected into the nonlinear feedback shift register (NFSR) are directly transmitted to the output, enabling the attacker to form low degree equations by observing the output difference [KR11a]. Solving the set of the equations facilitates the recovery of the internal state of the NFSR and subsequently the key. However, the presence of nonlinear hybrid cellular automata in the design of the cipher makes the formation of such equations almost infeasible. This claim is justified by the experimental results provided in the following section. Another notable difference of the M-NHCA from that of the NFSR in Grain is the absence of any feedback from the linear generator to the nonlinear one. This feedback is required to ensure large period length of the nonlinear register, and to provide mixing of the contents of linear and nonlinear registers. However, the M-NHCA $\mathcal{N}$ used here has maximum periodicity, and the bent function $h_{bent}(\cdot)$ uses six values from M-LHCA and two values from M-NHCA (as discussed in Section 2.3). In the design, the contents of linear and nonlinear registers are mixed in the initialization as well as keystream generation, therefore, such feedback is extraneous and subsequently discarded.

Choice of $h_l(\cdot)$ and $h_{bent}(\cdot)$. The function $h_l(\cdot)$ increases correlation immunity and resiliency whereas $h_{bent}(\cdot)$ provides high nonlinearity[1] [CS09], [MVV97]. In addition, the function $h_{bent}(\cdot)$ is designed to be a rotational symmetric one. This ensures that the occurrence of a fault is equiprobable for all the nonlinear terms in the bent function $h_{bent}(\cdot)$ in case of a faulty output. The lack of rotational symmetry of the filter function in Grain has already been exploited in [BMS12] which necessitates the use of such function to conceal the fault positions.

Choice of output function $h(\cdot)$. Recovery of the state bits by reverse engineering is prevented by the use of a combiner function. For this purpose, a Boolean function $h(\cdot)$ is selected as a sum of two parts, a nonlinear bent function $h_{bent}(\cdot)$ and a linear function $h_l(\cdot)$. The combiner function has nonlinearity 3840 and resiliency 4 that increase with iterations while expressed in terms of initial state bits. Because of incorporating rotational symmetric bent function $h_{bent}(\cdot)$, it strengthens the security of the cipher against attacks

[1]Bent function possesses the highest possible nonlinearity

like algebraic attack and fault attack etc.

4 Security Analysis

The proposed cipher aims to provide resistance against some related known attacks, in particular the fault attack which is the most threatening in the current scenario. This cipher is a Grain-like cipher having a nonlinear block ($\mathcal{N}$) and a linear block ($\mathcal{L}$). In the design of the cipher, both $\mathcal{N}$ and $\mathcal{L}$ are implemented with CA. $\mathcal{L}$ is a maximum period linear CA (M-LHCA). $\mathcal{N}$ is an M-NHCA. Wolfram proposed nonlinear CA with rule 30 as a better cryptographic primitive [Wol85], [Wol86]. However, in [MS91], Meier and Staffelbach have mounted an attack (MS attack) against Rule 30. The following analysis shows that the proposed cipher is secure against MS attack, and we also consider some general attacks on stream ciphers in this section. Before presenting analysis of the proposed cipher against MS attack, we describe the MS attack in detail in the following subsection.

4.1 Meier and Staffelbach (MS) Attack

In [MS91], the attack is a known plaintext attack where the keys are chosen as seed of the cellular automaton of size n (i.e. the size of the keys is n). The problem of cryptanalysis consists in determining the seed (or the keys) from the produced output sequence.

In [MS91], a general nonlinear cellular automata denoted by $\{s_1, s_2, \cdots, s_n\}$ of width $n = 2N + 1$ is considered. The state transition functions of all cells of the nonlinear cellular automata follow the rule 30. The site vector of the nonlinear CA at time step t is $(s_1^t, s_2^t, \cdots, s_{i-1}^t, s_i^t, s_{i+1}^t, \cdots, s_{2N+1}^t)$. The evolution of the i^{th} cell for N cycles is denoted by $\{s_i^t\}$ that is $\langle s_i^t, s_i^{t+1}, \cdots, s_i^{t+N}\rangle$. This bit-sequence is called the temporal sequence. The site vector, which is the key of this attack, forms a triangle along with the temporal sequence column (i.e. $\{s_i^t\}$) for N cycles as shown in Fig. 8. From the knowledge of

$$\begin{array}{ccccccc}
s_{i-N}^t & \cdots & s_{i-1}^t & s_i^t & s_{i+1}^t & \cdots & s_{i+N}^t \\
 & \cdots & s_{i-1}^{t+1} & s_i^{t+1} & s_{i+1}^{t+1} & \cdots & \\
 & & \vdots & \vdots & \vdots & & \\
 & & & \cdot & & & \\
 & & & s_i^{t+N} & & &
\end{array}$$

Figure 8: Determination of the seed

two adjacent columns in the triangle, that is, temporal sequence column (i.e. $\{s_i^t\}$) and right adjacent sequence column (i.e. $\{s_{i+1}^t\}$) or temporal sequence column (i.e. $\{s_i^t\}$) and left adjacent sequence column (i.e. $\{s_{i-1}^t\}$), one can determine the seed. Moreover, the knowledge of $\langle s_{i+1}^t, \cdots, s_{2N+1}^t\rangle$ together with the temporal sequence is sufficient to determine the triangle to the right of the temporal sequence column and the knowledge of $\langle s_1^t, \cdots, s_{i-1}^t\rangle$ together with the temporal sequence is sufficient to determine the triangle to the left of the temporal sequence column.

In [MS91], the site vector, the key of this attack, of the nonlinear CA at time step t is $(s_{i-N}^t, \cdots, s_{i-1}^t, s_i^t, s_{i+1}^t, \cdots, s_{i+N}^t)$. The bit-sequence of i^{th} cell is the known output sequence, where $i = N + 1$, that is the sequence of the middle cell and every cell of the null boundary nonlinear cellular automata follows Rule 30. The state transition function of Rule 30 is as follows:

$$s_i^{t+1} = s_{i-1}^t \oplus (s_i^t + s_{i+1}^t)$$

where s_i^t is the current state and s_i^{t+1} is the next state of the i^{th} cell. First, a random seed $\langle s_{i+1}^t, \cdots, s_{i+N}^t \rangle$ is generated.

In the completion forwards process, using the random seed and Rule 30 formula, s_{i+1}^{t+1}, s_{i+2}^{t+1}, $\cdots$, s_{i+N-1}^{t+1} can be easily computed as it is only the unknown item in the expression of Rule 30. In this way, the random seed together with temporal sequence column forms the right triangle as shown in Fig. 8. The above formula can be written in another way:

$$s_{i-1}^t = s_i^{t+1} \oplus (s_i^t + s_{i+1}^t)$$

In the completion backwards process, from the knowledge of right adjacent column (i.e. $\{s_{i+1}^t\}$) in the right triangle and the temporal sequence column (i.e. $\{s_i^t\}$), s_{i-1}^{t+N-1}, s_{i-1}^{t+N-2}, $\cdots$, s_{i-1}^t can be easily computed by the above expression since, it is only the unknown item in the expression. In this way, the knowledge of right adjacent column and the temporal sequence column can compute the left triangle of the temporal sequence column and eventually, determine the seed $\langle s_{i-N}^t, \cdots, s_{i-1}^t \rangle$ (completion backwards process [MS91]). Otherwise, generating a random seed $\langle s_{i-N}^t, \cdots, s_{i-1}^t \rangle$ together with temporal sequence column can form the left triangle as shown in Fig. 8 (completion forwards process [MS91]). The knowledge of left adjacent column (i.e. $\{s_{i-1}^t\}$) in the left triangle and the temporal sequence column $\{s_i^t\}$ can determine the seed $\langle s_{i+1}^t, \cdots, s_{i+N}^t \rangle$ (completion backwards process [MS91]).

Eventually, The CA is loaded with the computed seed $\langle s_{i-N}^t, \cdots, s_{i-1}^t, s_i^t, s_{i+1}^t, \cdots, s_{i+N}^t \rangle$ and produce the output sequence; the algorithm terminates if the produced sequence coincides with the known output sequence, otherwise, this process repeats for another choice of the random seed. There are 2^N ($\approx 2^{\frac{n}{2}}$) choices for random seed, so the required time complexity is $O(2^N)$(i.e.$O(2^{\frac{n}{2}})$).

4.2 Analysis against MS Attack

This work is briefly introduced in [MC18]. Here, the detailed proof of MS attack resistance of the synthesized M-NHCA is shown.

Let us consider a 3-neighborhood n-bit maximum period null-boundary LHCA denoted by $\{x_0, x_1, \cdots, x_{n-1}\}$ with rule vector $[d_0, d_1, \cdots, d_{n-1}]$, where,

$$d_l = \begin{cases} 0, & \text{if } x_l \text{ follows Rule 90} \\ 1, & \text{if } x_l \text{ follows Rule 150.} \end{cases}$$

Let the nonlinear functions $f_N(x_{j-2}^t, x_{j+2}^t)$ be injected at position j and $f_N(x_{k-2}^t, x_{k+2}^t)$ be injected at position k, where

$$f_N(x_{j-2}^t, x_{j+2}^t) = (x_{j-2}^t \cdot x_{j+2}^t),$$
$$f_N(x_{k-2}^t, x_{k+2}^t) = (x_{k-2}^t \cdot x_{k+2}^t)$$

and $k - j = 4$ as per criteria of nonlinear function injection into multiple inject points for producing M-NHCA $\mathcal{N}'$ as discussed in Section 2.2.1. The state transition functions of five neighboring cells of $\mathcal{N}'$ around the nonlinear inject positions j and k respectively, are as follows:

for j^{th} position:

$$x_{j-2}^{t+1} = x_{j-3}^t \oplus d_{j-2} \cdot x_{j-2}^t \oplus x_{j-1}^t \quad (1)$$

$$x_{j-1}^{t+1} = x_{j-2}^t \oplus d_{j-1} \cdot x_{j-1}^t \oplus x_j^t \oplus (x_{j-2}^t \cdot x_{j+2}^t) \quad (2)$$

$$x_j^{t+1} = x_{j-1}^t \oplus d_j \cdot x_j^t \oplus x_{j+1}^t \oplus d_j \cdot (x_{j-2}^t \cdot x_{j+2}^t) \oplus$$
$$((x_{j-3}^t \oplus d_{j-2} \cdot x_{j-2}^t \oplus x_{j-1}^t) \cdot (x_{j+1}^t \oplus d_{j+2} \cdot x_{j+2}^t \oplus x_{j+3}^t)) \quad (3)$$

$$x_{j+1}^{t+1} = x_j^t \oplus d_{j+1} \cdot x_{j+1}^t \oplus x_{j+2}^t \oplus (x_{j-2}^t \cdot x_{j+2}^t) \quad (4)$$

$$x_{j+2}^{t+1} = x_{j+1}^t \oplus d_{j+2} \cdot x_{j+2}^t \oplus x_{j+3}^t \quad (5)$$

for k^{th} position (in terms of j with $k = j + 4$):

$$x_{j+2}^{t+1} = x_{j+1}^t \oplus d_{j+2} \cdot x_{j+2}^t \oplus x_{j+3}^t \tag{6}$$

$$x_{j+3}^{t+1} = x_{j+2}^t \oplus d_{j+3} \cdot x_{j+3}^t \oplus x_{j+4}^t \oplus (x_{j+2}^t \cdot x_{j+6}^t) \tag{7}$$

$$\begin{aligned} x_{j+4}^{t+1} &= x_{j+3}^t \oplus d_{j+4} \cdot x_{j+4}^t \oplus x_{j+5}^t \oplus d_{j+4} \cdot (x_{j+2}^t \cdot x_{j+6}^t) \oplus \\ &((x_{j+1}^t \oplus d_{j+2} \cdot x_{j+2}^t \oplus x_{j+3}^t) \cdot (x_{j+5}^t \oplus d_{j+6} \cdot x_{j+6}^t \oplus x_{j+7}^t)) \end{aligned} \tag{8}$$

$$x_{j+5}^{t+1} = x_{j+4}^t \oplus d_{j+5} \cdot x_{j+5}^t \oplus x_{j+6}^t \oplus (x_{j+2}^t \cdot x_{j+6}^t) \tag{9}$$

$$x_{j+6}^{t+1} = x_{j+5}^t \oplus d_{j+6} \cdot x_{j+6}^t \oplus x_{j+7}^t \tag{10}$$

where $\langle x_0^t, x_1^t, \cdots, x_{n-1}^t \rangle$ is the site vector of M-NHCA $\mathcal{N}'$ at time step t and $\langle x_0^{t+1}, x_1^{t+1}, \cdots, x_{n-1}^{t+1} \rangle$ is the site vector at time step $t+1$. The state transition functions for other cells that is $x_l^{t+1}, 0 \leq l \leq j-1$ and $j+7 \leq l \leq n-1$ can be formed by 90/150 rules. It is noted that (5) and (6) shown above are the same equation, because right of right neighbor of j^{th} cell and left of left neighbor of k^{th} cell denote the same cell position.

Suppose, we are given the output sequence $\{x_i^t\}$ (i.e. the temporal sequence $\{x_{j+3}^t\}$) up to the unicity distance N as shown in Table 2, where $i = j + 3$ and $i = k - 1$ since $k - j = 4$. Now, our aim is to determine the seed $\langle x_0^t, x_1^t, \cdots, x_{i-1}^t, x_i^t, x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ from the knowledge of given output sequence $\{x_i^t\}$. Here, the site vector $\langle x_0^t, x_1^t, \cdots, x_{n-1}^t \rangle$ forms two triangles (left and right) across the output sequence column. The right triangle is determined in the completion forwards process and the left triangle is determined in the completion backwards process.

Table 2: Computing seed for M-NHCA $\mathcal{N}'$

x_0^t	$\cdots$	x_{i-6}^t $*$	x_{i-5}^t $*$	$\cdots$	x_{i-1}^t	x_i^t	x_{i+1}^t $*$	$\cdots$ $***$	x_{n-1}^t $*$
x_0^{t+1}	$\cdots$	$*$	$*$	$\cdots$	x_{i-1}^{t+1}	x_i^{t+1}	x_{i+1}^{t+1}	$\cdots$	x_{n-1}^{t+1}
	$\cdots$	$*$	$*$	$\cdots$	x_{i-1}^{t+2}	x_i^{t+2}	x_{i+1}^{t+2}	$\cdots$	
	$\cdots$	$*$	$*$	$\cdots$	x_{i-1}^{t+3}	x_i^{t+3}	x_{i+1}^{t+3}	$\cdots$	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
		$*$	$*$	$\cdots$	.	.	.	$\cdots$	
			$*$	$\cdots$	.	.	.	$\cdots$	
				$\cdots$	.	.	.	$\cdots$	
					x_{i-1}^{t+N-1}	.	x_{i+1}^{t+N-1}		
						x_i^{t+N}			

’*’ represents "guess" value

We choose a random seed $\langle x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ out of $2^{n-(i+1)}$ possibilities. Equation (7) can be written in another way:

$$x_{j+2}^t = x_{j+3}^{t+1} \oplus d_{j+3} \cdot x_{j+3}^t \oplus x_{j+4}^t \oplus (x_{j+2}^t \cdot x_{j+6}^t) \tag{11}$$

Now, x_{j+2}^t can be determined from (11) with probability $\frac{1}{2}$. In the completion forwards process (i.e. left to right approach), x_{j+4}^{t+1}, x_{j+5}^{t+1} can be computed using (8) and (9) respectively, in the 2nd rule set. x_{j+6}^{t+1}, x_{j+7}^{t+1}, $\cdots$,x_{n-1}^{t+1} can be computed as per 3-neighborhood 90/150 rule. For next time step (i.e at time step $t+2$) we can compute all above values again using the 2nd rule set. In this way, right triangle of the temporal sequence column ($i.e\{x_i^t\}$), shown in Table 2, can be determined. Here, the only knowledge of right adjacent column in the right triangle together with temporal sequence column can not determine the seed $\langle x_0^t, \cdots, x_{i-1}^t \rangle$. Equation (5) can be written in another way:

$$x_{j+1}^t = x_{j+2}^{t+1} \oplus d_{j+2} \cdot x_{j+2}^t \oplus x_{j+3}^t \tag{12}$$

The column $\{x_{j+1}^t\}$ can be computed from (12). Similarly, (4) can be written in another way:

$$x_j^t = x_{j+1}^{t+1} \oplus d_{j+1} \cdot x_{j+1}^t \oplus x_{j+2}^t \oplus (x_{j-2}^t \cdot x_{j+2}^t) \tag{13}$$

The column $\{x_j^t\}$ can only be computed from (13) if the column $\{x_{j-2}^t\}$ (i.e.$\{x_{i-5}^t\}$) is chosen at random out of 2^{j+1} possibilities. Similarly, the column $\{x_{j-1}^t\}$ can only be computed from (3) of the 1st rule set if the column $\{x_{j-3}^t\}$ (i.e.$\{x_{i-6}^t\}$) is chosen at random out of 2^j possibilities, because $\{x_{j-3}^t\}$ is unknown. The columns $\{x_{j-4}^t\}, \{x_{j-5}^t\}, \cdots, \{x_0^t\}$ can be computed as per 3-neighborhood 90/150 rule. Here, each column is computed by bottom-up approach. In this way left triangle of the temporal sequence column ($i.e\{x_i^t\}$) can be formed (completion backwards process) and hence, the seed $\langle x_0^t, \cdots, x_{i-1}^t \rangle$ can be determined.

Eventually, the CA is loaded with the computed seed $\langle x_0^t, \cdots, x_{i-1}^t, x_i^t, x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ and produce the output sequence; the algorithm terminates if the produced sequence coincides with the given temporal sequence, otherwise, this process repeats for another choice of random seed $\langle x_{i+1}^t, \cdots, x_{n-1}^t \rangle$.

The random seed $\langle x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ can be chosen with $2^{n-(i+1)}$ possibilities. Since, x_{j+2}^t is determined from (7) with probability $\frac{1}{2}$, therefore, for the column $j+2$, $\frac{n-(i+1)}{2}$ values can be computed deterministically and other $\frac{n-(i+1)}{2}$ values can be chosen randomly with $2^{\frac{n-(i+1)}{2}}$ possibilities. The column $j-2$ is chosen at random out of 2^{j+1} possibilities. The column $j-3$ is chosen at random out of 2^j possibilities. Therefore, the required time complexity is:

$$\begin{aligned} 2^{n-(i+1)}.2^{\frac{n-(i+1)}{2}}.2^{j+1}.2^j &= 2^{\frac{3}{2}(n-i-1)}.2^{2j+1} \\ &= 2^{n+\frac{3}{4}(n-9)} \end{aligned}$$

where $j = i-3$ and $i = \frac{n-1}{2}$, the middle cell position of the CA. Hence, the required time is greater than 2^n (reqd. for exhaustive search) for $n > 9$.

Following the similar approach, we can determine the seed $\langle x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ from the given output sequence $\{x_i^t\}$ upto the unicity distance N by guessing the seed $\langle x_0^t, \cdots, x_{i-1}^t \rangle$ out of 2^i possibilities as shown in Table 3. Here, the left triangle is determined in the completion forwards process and the right triangle is determined in the completion backwards process. Equation (7) can be written in another way:

$$x_{j+4}^t = x_{j+3}^{t+1} \oplus x_{j+2}^t \oplus d_{j+3} \cdot x_{j+3}^t \oplus (x_{j+2}^t \cdot x_{j+6}^t) \tag{14}$$

In the completion backwards process, the column $\{x_{j+4}^t\}$ (i.e. $\{x_{i+1}^t\}$) can only be computed from (14) if the column $j+6$ is chosen at random out of 2^{n-k} possibilities and similarly, the column $\{x_{j+5}^t\}$ can only be computed from (8) if the column $j+7$ is chosen at random out of 2^{n-k-1} possibilities. Hence, the seed $\langle x_{i+1}^t, \cdots, x_{n-1}^t \rangle$ can be determined. Therefore, the required time complexity is:

$$2^i.2^{n-k}.2^{n-k-1} = 2^{i+2n-2k-1} = 2^{n+\frac{n-5}{2}}$$

where $k = j+4 = i+1$ and $i = \frac{n-1}{2}$, the middle cell position of the CA. Hence, the required time is greater than 2^n (reqd. for exhaustive search) for $n > 5$.

Table 3: Computing seed for M-NHCA $\mathcal{N}'$

x_0^t $*$	$\cdots\cdots$ $***$	x_{i-1}^t $*$	x_i^t	x_{i+1}^t	x_{i+2}^t	x_{i+3}^t $*$	x_{i+4}^t $*$	$\cdots$	x_{n-1}^t
x_0^{t+1}	$\cdots$	x_{i-1}^{t+1}	x_i^{t+1}	x_{i+1}^{t+1}	x_{i+2}^{t+1}	$*$	$*$	$\cdots$	x_{n-1}^{t+1}
	$\cdots$	x_{i-1}^{t+2}	x_i^{t+2}	x_{i+1}^{t+2}	x_{i+2}^{t+2}	$*$	$*$	$\cdots$	x_{n-1}^{t+2}
	$\cdots$	x_{i-1}^{t+3}	x_i^{t+3}	x_{i+1}^{t+3}	x_{i+2}^{t+3}	$*$	$*$	$\cdots$	
	$\cdots$	x_{i-1}^{t+4}	x_i^{t+4}	x_{i+1}^{t+4}	x_{i+2}^{t+4}	$*$	$*$	$\cdots$	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
		.	.	.	.	$*$	$*$		
		x_{i-1}^{t+N-1}	.	x_{i+1}^{t+N-1}		$*$			
			x_i^{t+N}						

'*' represents "guess" value

Note that the synthesized M-NHCA is a null boundary CA, and there is one tap bit (i.e. i-th cell) from which the output sequence $\{x_i^t\}$ is given. Nonlinearity injection in both sides of the tap bit can resist MS attack for $n > 9$.

In the proposed cipher, the nonlinear block is designed with a synthesized 128-bit M-NHCA. The set of tap bits taken from the nonlinear block is $\{b_{16}, b_{32}, b_{47}, b_{67}, b_{80}, b_{96}, b_{112}\}$ for the combiner function (as discussed in Section 2.3). Nonlinearity is injected in both sides of each tap bit to resist MS attack. The set of injection points has already been shown in Section 2.2. The distance between 1st and 3rd tap bits of a sequence of three consecutive tap bits is maintained as greater than 9. The distance for the sequence $\langle b_{16}, b_{32}, b_{47}\rangle$ is $47 - 16 + 1 = 32$, and the distance for the sequence $\langle b_{32}, b_{47}, b_{67}\rangle$ is $67 - 32 + 1 = 36$, and so on. Thus, the M-NHCA can resist MS attack even if the tap bits are known.

4.3 Algebraic Cryptanalysis

Algebraic cryptanalysis depends on constructing a probabilistic pattern of the outputs to distinguish the cipher from a random permutation and solving low degree equations from them. As M-LHCA involves only linear terms, a combiner Boolean function $h(\cdot)$ constructed out of it is immediately susceptible to algebraic attacks compromising its security. This can be prevented by introducing nonlinear function in the design along with the M-LHCA and $h(\cdot)$. This is achieved by the nonlinear transition function of the M-NHCA which causes the algebraic degree of the output expressed in terms of initial state bits to increase. Table 4 shows d-monomial characteristics along with Algebraic Degree and number of state variables of the output of the cipher with iterations; where the output bit of the cipher depends on 128 state variables with Algebraic Degree 6 at iteration 8 and the number of nonlinear terms in the output expression is 297201 at iteration 8 and these increase with iterations. Hence, after 128 clock cycles (i.e. initialization phase), the output bit will be dependent on almost 256 state variables with higher Algebraic Degree. The increase of number of nonlinear terms and the Algebraic degree of a cipher also increases the attack complexity. Therefore, from the result of Table 4, it is expected that the recovery of the internal state from the output is beyond practical measure.

4.4 Linear Approximation and Correlation Attack

The linear cryptanalysis technique depends on approximating the output with an affine function. Its success is directly related to the biases present in the output bits of the cipher. The design choice of the cipher precludes such possibility as the output of both $\mathcal{L}$ and $\mathcal{N}$

Table 4: Different characteristics of the cipher output

Itr #	Rsly	Alg. Deg.	# of Vars.	Deg-1	Deg-2	Deg-3	Deg-4	Deg-5	Deg-6	Deg-7	Deg-8
0	4	4	13	5	24	32	24	0	0	0	0
1	7	6	40	13	154	519	1057	387	35	0	0
2	14	5	50	16	335	1646	4528	531	0	0	0
3	14	6	64	17	462	2734	9146	2964	250	0	0
4	25	6	90	29	1056	9267	46644	9869	506	0	0
5	29	6	99	35	1217	11542	62096	12432	578	0	0
6	32	6	102	37	1250	12183	63930	8554	292	0	0
7	36	6	126	41	2343	30881	230041	34172	1140	0	0
8	34	6	128	45	2402	31832	239044	23386	537	0	0
9	37	6	130	45	2521	34519	264520	26458	681	0	0
10	33	8	152	50	4108	72617	775540	638331	808594	592230	21613

# of Deg-3 monomials at iteration #6: 12183 Rsly ≡ Resiliency

are balanced. Moreover, nonlinearity is incorporated to prevent the affine approximation. This is accomplished with a nonlinear sequence generator $\mathcal{N}$ and a combiner function $h(\cdot)$. The nonlinearity of the entire state of $\mathcal{N}$ can be ascertained by studying the nonlinearity of the injection points. Such a study is furnished in Table 5 which shows the increase in nonlinearity of the injection points with number of iterations. Moreover, the final combiner function $h(\cdot)$ has nonlinearity 3840 that also increases over time while expressed in terms of initial state bits.

Correlation attack is a class of known IV attack which exploits the statistical weakness of the underlying combiner function $h(\cdot)$. It works by exploring the dependency of the output bit-stream on IV due to poor choice of the combiner function. Security against such attacks requires a careful choice of a Boolean function providing certain degree of immunity against correlation. This is obtained by adding 5 linear terms by the function $h_l(\cdot)$ to the combiner function along with $h_{bent}(\cdot)$ resulting in a correlation immunity of 4. Moreover, the addition of $h_l(\cdot)$ makes the combiner function a balanced function and makes it balanced with iterations. While evolving CA, each variable of the combiner function $h(\cdot)$ is updated based on some linear terms and some nonlinear terms of initial state bits of both the linear CA and nonlinear CA. Therefore, the combiner function $h(\cdot)$ gets more linear terms of CA states with iterations and accordingly the number of linear terms in the combiner function $h(\cdot)$ increases during initialization. Hence, the resiliency of $h(\cdot)$ increases with iterations. This has been shown in Table 4, where the number of Deg-1 monomials and resiliency are increased with iterations. Thus, due to the faster growth of resiliency of the output bit of the cipher, it is expected that this cipher is resistant against Correlation Attack.

Table 5: Nonlinearity of $\mathcal{N}$ for various iterations

Itr #	Nonlinear function injection points													
	13	17	29	33	44	48	64	68	77	81	93	97	109	113
1	4	4	48	48	8	8	16	48	16	48	48	8	48	8
2	32	64	64	128	32	128	128	64	64	64	256	128	64	64
3	128	128	128	128	128	768	512	256	1536	512	512	256	256	1024
4	1536	768	3072	512	384	1024	1536	256	3072	512	512	512	768	2048

4.5 Statistical Analysis

Statistical analysis is another powerful cryptanalysis tool that is used to cryptanalyze all kind of symmetric ciphers. To design a cipher, it is of utmost importance to make sure that the generated output stream has negligible bias which cannot be exploited by any adversary to mount an attack. For this purpose, a statistical test suite is developed by National Institute of Standards and Technology (NIST) that is known as NIST-statistical test suite. 100 bit-streams with each stream of 1,000,000 bits are generated from the proposed cipher and stored in a data file, and then the data file is fed to NIST test suite. The generated bit-streams pass all tests of randomness property by NIST test suite.

4.6 Analysis against Fault Attack

Fault attack has gained considerable attention from the research community. As mentioned earlier, the eSTREAM winners like Grain, Trivium, have been shown to be vulnerable against such attacks [HS04], [BMS12], [HR08] irrespective of their theoretical security. These attacks were successful due to the low diffusion rate of the feedback shift register. However, the presence of cellular automata and the use of rotational symmetric bent function $h_{bent}(\cdot)$ in the cipher nullifies such scenario as CA are known to possess higher diffusion rate than FSRs.

In this work, we have studied the effects of injecting single-bit fault into various locations of $\mathcal{N}$ and $\mathcal{L}$. After the initialization phase, we run the cipher for a target cycle T to inject fault. We refer to this point as the *base point*. Let the state of the cipher be $\langle b_0^T, b_1^T, \cdots, b_{127}^T, s_0^T, s_1^T, \cdots, s_{127}^T \rangle$ at the base point T. Single-bit fault is injected either in the NHCA $\mathcal{N}$ or LHCA $\mathcal{L}$ at time T. Further, we denote the output difference of faulty and fault-free keystream at iteration t after the base point by δ^t, i.e. $\delta^t = z^t_{fault-free} + z^t_{faulty}$. δ^t is used to obtain internal state of the cipher (i.e. $b_0^T, b_1^T, \cdots, b_{127}^T, s_0^T, s_1^T, \cdots, s_{127}^T$) at the base point T. After injecting single-bit fault at the base point T, we have run the cipher for t cycles (say, $t = 10$). As a result, only 13 bits of $\mathcal{N}$ at the base point T can be obtained and no bits of $\mathcal{L}$ can be obtained as shown in Table 6.

The $h_{bent}(\cdot)$ function contains 80 nonlinear terms with 8 variables and maximum degree 4, and each variable appears in 30 nonlinear terms among 80 nonlinear terms. Moreover, the design of the cipher increases the degree, number of variables and number of non-linear terms in the output expression with iterations as shown in Table 4, so δ^t will contain large number of nonlinear terms with higher degree with iterations and hence, it is hardly possible to obtain all 256 bits at the base point (i.e., $b_0^T, b_1^T, \cdots, b_{127}^T, s_0^T, s_1^T, \cdots, s_{127}^T$) and to recover the *key* from this internal state. Thus, the design is expected to be resistant against fault attack.

Table 6: Fault location vs. NHCA bits obtained

Fault location	Itr #	NHCA bits	Fault location	Itr #	NHCA bits	Fault location	Itr #	NHCA bits
b_{27}	4	b_{32}	b_{46}	4	b_{43}	b_{75}	4	b_{80}
b_{31}	4	b_{28}	b_{50}	2	b_{47}	b_{79}	4	b_{76}
b_{31}	6	b_{36}	b_{62}	4	b_{67}	b_{83}	2	b_{80}
b_{35}	2	b_{32}	b_{66}	4	b_{63}	b_{107}	4	b_{112}
b_{42}	4	b_{47}	b_{66}	5	b_{71}	b_{111}	5	b_{108}
b_{46}	3	b_{51}	b_{70}	2	b_{67}	b_{115}	2	b_{112}

5 Hardware Implementation

In this section, we provide hardware results of the proposed stream cipher (Fig. 7), and the comparison with Grain-128 [HJMM06] and other Grain-like CA based ciphers [KR11b], [GR15]. The ciphers shown in Table 7 are all implemented on Xilinx Spartan 3 XC3S200-4FT256 device using Xilinx 14.2 synthesis tool. The result shows that Grain-128 is hardware efficient than the other ciphers while the throughput to area ratio (i.e. efficiency) of the proposed cipher is 1.434 which is better than grain-128. The proposed cipher achieves two times speedup in initialization than Grain-128.

Table 7: Comparison with Grain-128 and other CA based ciphers

Ciphers	No. of 4-input LUTs	Maximum clock frequency (MHz)	Maximum throughput (Mbps)	Area (slices)	Throughput/ Area (Mbps/slice)	Setup (cycles)
Grain-128	27	178.094	178.094	227	0.785	256
NOCAS	761	365.764	365.764	514	0.711	64
CASca	269	238.892	238.892	137	1.744	128
Our cipher	325	236.630	236.630	165	1.434	128

Table 8 shows the performance variation with several nonlinear function injection points. It is noticed that the efficiency increases with decrease of the number of injection points. The result shown in Table 8 gives some directions how the number of injection points governs the performance of the cipher. To prevent MS attack, maintaining injection points in both sides of the tap points is necessary that we have shown the proof in Section 4.2. So the decrease of injection points disturbs this property and makes the cipher vulnerable against MS attack.

Table 8: Cipher-performance with nonlinear function injection points

No. of injection points	No. of 4-input LUTs	Maximum clock frequency (MHz)	Maximum throughput (Mbps)	Area (slices)	Throughput/ Area (Mbps/slice)
14	325	236.630	236.630	165	1.434
12	325	237.869	237.869	165	1.442
10	325	237.869	237.869	165	1.442
8	323	237.869	237.869	164	1.450
6	322	237.869	237.869	163	1.459
4	318	237.869	237.869	161	1.477

6 Conclusion

In this paper, we have studied and analyzed the cryptographic properties of maximum period nonlinear CA. The CA is synthesized by injecting nonlinear functions in multiple inject points. Use of these nonlinear CA makes the cipher resistant against MS attack. Unlike Grain-128, our design eliminates the feedback function from linear to nonlinear block. This elimination and the use of rotational symmetric bent function used in the combiner make the cipher strong against fault attack. This CA based cipher is scalable, and very well suited for hardware which is evident from the FPGA implementation of the design.

References

[BMS12] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on the grain family of stream ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 122–139, 2012.

[CM96] Kevin Cattell and Jon C. Muzio. Synthesis of one-dimensional linear hybrid cellular automata. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 15(3):325–335, 1996.

[CRNC97] P. P. Chaudhuri, D. Roy Chowdhury, S. Nandi, and S. Chattopadhyay. *Additive Cellular Automata: Theory and Applications.* IEEE Computer Socity press, 1997.

[CS09] Thomas W. Cusick and Pantelimon Stanica. *Cryptographic Boolean Functions and Applications.* Academic Press, 2009.

[DR13] Sourav Das and Dipanwita Roy Chowdhury. CAR30: A new scalable stream cipher with rule 30. *Cryptography and Communications*, 5(2):137–162, 2013.

[DR14] Sourav Das and Dipanwita Roy Chowdhury. CASTREAM: A high-speed, secure stream cipher suitable for both hardware and software. *J. Cellular Automata*, 9(2-3):153–166, 2014.

[GR15] Shamit Ghosh and Dipanwita Roy Chowdhury. CASca: A CA based scalable stream cipher. In *Mathematics and Computing - 2nd International Conference on Mathematics and Computing 2015*, pages 95–105, 2015.

[GSSR14] Shamit Ghosh, Abhrajit Sengupta, Dhiman Saha, and Dipanwita Roy Chowdhury. A scalable method for constructing non-linear cellular automata with period $2^n - 1$. In *Cellular Automata - 11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Krakow, Poland, September 22-25, 2014. Proceedings*, pages 65–74, 2014.

[HJMM06] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *Information Theory, 2006 IEEE International Symposium*, pages 1614–1618, 2006.

[HR08] Michal Hojsík and Bohuslav Rudolf. Floating fault analysis of trivium. In *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, pages 239–250, 2008.

[HS04] Jonathan J. Hoch and Adi Shamir. Fault analysis of stream ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 240–253, 2004.

[KR11a] Sandip Karmakar and Dipanwita Roy Chowdhury. Fault analysis of grain-128 by targeting NFSR. In *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, pages 298–315, 2011.

[KR11b] Sandip Karmakar and Dipanwita Roy Chowdhury. NOCAS : A nonlinear cellular automata based stream cipher. In *17th International Workshop on Cellular Automata and Discrete Complex Systems, Automata 2011, Center for*

Mathematical Modeling, University of Chile, Santiago, Chile, November 21-23, 2011, pages 135–146, 2011.

[MC18] Swapan Maiti and Dipanwita Roy Chowdhury. Achieving better security using nonlinear cellular automata as a cryptographic primitive. In *Mathematics and Computing - 4th International Conference, ICMC 2018, Varanasi, India, January 9-11, 2018, Revised Selected Papers*, pages 3–15, 2018.

[MGR17] Swapan Maiti, Shamit Ghosh, and Dipanwita Roy Chowdhury. On the security of designing a cellular automata based stream cipher. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, pages 406–413, 2017.

[MS91] Willi Meier and Othmar Staffelbach. Analysis of pseudo random sequence generated by cellular automata. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, pages 186–199, 1991.

[MVV97] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* Boca Raton: CRC Press, 1997.

[Wol85] Stephen Wolfram. Cryptography with cellular automata. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pages 429–432, 1985.

[Wol86] Stephen Wolfram. Random sequence generation by cellular automata. In *Advances in Applied Mathematics, vol-7*, pages 123–169, 1986.